UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.
08/959,149	10/28/97	LIMPRECHT R	3382-47280

EXAMINER	

TM02/1219

KLARQUIST SPARKMAN CAMPBELL
LEIGH & WHINSTON
ONE WORLD TRADE CENTER SUITE 1600
121 S W SALMON STREET
PORTLAND OR 97204-2988

ART UNIT	PAPER NUMBER
2151	

DATE MAILED: 12/19/00

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner of Patents and Trademarks

Office Action Summary

Application No.	Applicant(s)
08/959,149	Limprecht, et al
Examiner	**Group Art Unit**
S. Lao	2151

☒ Responsive to communication(s) filed on *Oct 4, 2000*

☐ This action is **FINAL.**

☐ Since this application is in condition for allowance except for formal matters, **prosecution as to the merits is closed** in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11; 453 O.G. 213.

A shortened statutory period for response to this action is set to expire *3* month(s), or thirty days, whichever is longer, from the mailing date of this communication. Failure to respond within the period for response will cause the application to become abandoned. (35 U.S.C. § 133). Extensions of time may be obtained under the provisions of 37 CFR 1.136(a).

Disposition of Claim

☒ Claim(s) *1-24* is/are pending in the applicat

Of the above, claim(s) _____ is/are withdrawn from consideration

☐ Claim(s) _____ is/are allowed.

☒ Claim(s) *1-24* is/are rejected.

☐ Claim(s) _____ is/are objected to.

☐ Claims _____ are subject to restriction or election requirement.

Application Papers

☐ See the attached Notice of Draftsperson's Patent Drawing Review, PTO-948.

☐ The drawing(s) filed on _____ is/are objected to by the Examiner.

☐ The proposed drawing correction, filed on _____ is ☐ approved ☐ disapproved.

☐ The specification is objected to by the Examiner.

☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. § 119

☐ Acknowledgement is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d).

☐ All ☐ Some* ☒ None of the CERTIFIED copies of the priority documents have been

☐ received.

☐ received in Application No. (Series Code/Serial Number) _____.

☐ received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

*Certified copies not received: _____

☐ Acknowledgement is made of a claim for domestic priority under 35 U.S.C. § 119(e).

Attachment(s)

☒ Notice of References Cited, PTO-892

☒ Information Disclosure Statement(s), PTO-1449, Paper No(s). *9, 11, 12*

☐ Interview Summary, PTO-413

☐ Notice of Draftsperson's Patent Drawing Review, PTO-948

☐ Notice of Informal Patent Application, PTO-152

--- SEE OFFICE ACTION ON THE FOLLOWING PAGES ---

DETAILED ACTION

1. Claims 1-24 are pending. This action is in response to the amendment filed 10/4/2000. Applicant has amended claim 1 and added claims 21-24.

2. The text of those sections of Title 35, U.S. Code not included in this action can be found in a prior Office action.

3. Claims 1, 3-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over The Common Object Request Broker: Architecture and Specification CORBA (Revision 2.0) in view of Ratner et al (U S Pat. 5,889,957) and Steinman ("Incremental State Saving in SPEEDS Using C++").

As to claim 21, CORBA teaches (chapter 4, pages 12-16) server applications (servers, applications), executing (invoke) an application component (object) under control of an operating service (ORB), the application component having a state (context) and function code (method) for performing work responsive to a call (invoke method) from a client (client), destroying the state by the operating service (delete context object by CORBA::CTX_DELETE()). It is noted that a destroyed state in CORBA is not persistent.

CORBA does not teach (1) the step of maintaining, (2) destroying is in response to an indication from application component.

As to (1), Steinman teaches (SPEEDS system) maintaining an object state (v1) in main memory between method invocations (between events/messages/method calls, by delta exchange method), see sections 3, 4. Since CORBA as modified and Steinman address object lifetime management (creation/deletion), it would have been obvious to combine the teachings. It is noted that Steinman maintains object state without requiring an indication that work is complete.

As to (2), Ratner teaches in response to an indication that work is complete (API: DIALOG_END), the operating service (operating system) cleans up system resources. (Col. 10, lines 24-26). It would have been obvious that CORBA cleans up system resources

(destroy object states) in response to an indication and without action by the client. The motivation to combine includes that CORBA requires transient states/contexts (page 4-13) and Ratner provides a transient mechanism (clean up resource after a session ends).

As to claim 1, CORBA teaches (chapter 4, pages 12-16) server applications (servers, applications), executing (invoke) an application component (object) under control of an operating service (ORB), the application component having a state (context) and function code (method) for performing work responsive to a call (invoke method) from a client (client), destroying the state by the operating service (delete context object by CORBA::CTX_DELETE()).

CORBA does not teach the steps of (1) providing and (2) maintaining, nor (3) destroying is in response to an indication and (4) without action by the client.

As to (1), Ratner teaches client/server resource management, including providing an interface for the operating service to receive (APIs) an indication that work is complete (SERVERCLASS_DIALOG_END). The API is sent from an application component (break a session by the server). See col. 6, line 66 - col. 7, line 8; col. 4, lines 36-37. It is noted that completion of work is represented by the end of a session in Ratner.

As to (3)-(4), Ratner teaches in response to an indication that work is complete (API: DIALOG_END) and without action by the client (API called by the server), the operating service (operating system) cleans up system resources. (Col. 10, lines 24-26). It would have been obvious that CORBA cleans up system resources (destroy object states) in response to an indication and without action by the client. The motivation to combine includes that CORBA requires transient states/contexts (page 4-13) and Ratner provides a transient mechanism (clean up resource after a session ends).

As to (2), Steinman teaches (SPEEDS system) maintaining an object state (v1) in main memory between method invocations (between events/messages/method calls, by delta exchange method), see sections 3, 4. Since CORBA as modified and Steinman address object lifetime management (creation/deletion), it would have been obvious to combine the teachings. It is noted that Steinman maintains object state without requiring an indication that work is complete.

As to claim 3, CORBA as modified teaches (Steinman) resetting the state (restore state by calling exchange again, section 4).

As to claim 4, CORBA as modified teaches (Ratner) upon a next return following the indication in that a DIALOG_END following a previous FeOK message initiates the state clean up. Col. 10, lines 8-26.

As to claim 5, note discussion of claims 1 and 4 except for the following limitations.

CORBA teaches (discussion of claim 1) run-time service (ORB) for executing and destroying. CORBA further teaches (chapter 2, page 9, section 2.1.11) instance creation service (object activation), client request (request), return a reference (generate object reference). Typically in CORBA, a client calls object/component functions indirectly by calling a stub which is a form of object adapter to initiate work (invoke object) through the run-time service (ORB, including object adapter) using the reference (stub).

As to claim 6, CORBA as modified teaches (Ratner) calls from client to commit or abort (DIALOG_END, DIALOG_ABORT, callable by client) (col. 6, line 66 - col. 7, line 8; col. 4, lines 36-37). Note rejection of claim 1 for motivation to combine.

As to claim 7, CORBA as modified teaches (Ratner) application component initiates the indication before returning (server calls DIALOG_END) (see discussion of claim 1 with respect to Ratner) and destroying state immediately on return (DIALOG_END causes the operating system to clean up system resources) (Col. 10, lines 24-26).

As to claim 8, CORBA teaches component context (context object) associated with the application component (object) (see discussion of claim 1 with respect to CORBA). CORBA provides an interface having member functions to manipulate the component context (context interfaces, pages 4-13 to 4-16, such as to set a value, section 4.6.2). Therefore, including an interface to initiate an indication (to set a value) would have been obvious.

As to claims 9-10, CORBA as modified teaches (Ratner) teaches transaction environment (transaction), and calling component's member function (server class) to abort (SERVERCLASS_DIALOG_ABORT). Since CORBA provides a set of context interfaces to manipulate the component context (discussion of claim 8), it would have been obvious

to include a context interfaces for abort and for commit since these are also functions manipulate the component context.

As to claim 11, holding a reference and releasing a reference are conventional means for object creation and destruction.

As to claim 12, note the discussion of claim 3.

As to claim 13, note discussion of claims 1 and 5, and note the equivalence of discarding (claim 13) / destroying (claims 1, 5). CORBA further teaches encapsulating function code (object method) and a processing state for the work in a component (context object), providing a reference (object reference) through an operating service (CORBA) for a client program to call the function code of the component to initiate processing (invoke method) (see discussion of claim 1 with respect to CORBA).

As to claims 14-17, note discussion of claims 4, 8-10, respectively. Further, CORBA teaches context object (context object). As to integration interface for receiving a call of component/object, note discussion of claim 1 with respect to step providing an interface.

As to claim 18, it is basically a program product claim of claim 5. Note rejection of claim 5.

As to claims 19-20, note discussion of claims 11-12. The factory mechanism of CORBA produces component/object instance and its pointer. When an object is reused, its state is typically reset/reinitialized.

4. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over CORBA in view of Ratner et al and Steinman as applied to claim 1 and further in view of Bishop (U S Pat. 7,765,174).

As to claim 2, Bishop teaches destroying the state (delete object) while retaining a client reference to the object (maintain weak references to object). See col. 1, lines 28-44. Since CORBA and Bishop address object deallocation, it would have been obvious to combine the teachings.

5. Claims 22-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over CORBA in view of Ratner et al and Steinman as applied to claims 1, 3, 4 and further in view of author admitted prior art APA (pages 1, line 10 - page 5, line 10).

As to claim 22, it is covered by claims 1 and 4 except for work data state and persistently maintaining the work data state. CORBA further teaches state/context of a component/object comprises component data state (application portion of environment information) and work data state (client portion of environment information), and context can be maintained persistently or transiently. (page 4-12 to 4-13). CORBA does not teach the work data state is maintained persistently.

APA teaches work data state (user data/state) is maintained persistently (stored in secondary storage). See page 2, lines 10-21. Since CORBA requires persistent maintenance mechanism and APA provides one (secondary storage), it would have been obvious to combine the teachings.

As to claim 23, note rejection of claim 3.

As to claim 24, it is a program product claim of claim 22. Note claim 22 for rejection.

6. Applicant's arguments filed 10/4/2000 have been considered but are moot in view of the new ground(s) of rejection.

A. Regarding an interface for the operating service to receive an indication from the application component that work is complete as recited in claim 1, applicant argued that applicant's server application component calls either "IObjectContext::SetComplete()" or "IObjectContext::SetAbort()" to make the indication, (page 4, last para. - page 5, first and third para.). While they may be disclosed, these interfaces are not claimed. The argument is thus not persuasive.

B. Regarding the Steinman reference, applicant argued (1) Steinman fails to teach object indicates to the Speeds operating system that work is complete as recited in claim 1 (page 5, last para.), (2) Steinman does not teach resetting to object's initial post-creation state because the state reverse in Steinman is for a single event, and (3) Steinman teaches

away from the present invention because the delta exchange method adversely affect scalability due to its space overhead of the rollback queues.

The examiner disagrees. As to (1), Steinman is not relied on to teach an interface for the operating service to receive an indication from the application component that work is complete as recited in claim 1, which is met by the other references relied on. See discussion of claim 1 for detail.

As to (2), Steinman resets an object state to the previous state before an event which is produced by a method invocation. See sections 3 and 4. Steinman does not limit the starting point or the duration of such an event/invocation. Obviously the event includes but is not limited to a first event and the corresponding previous state includes but is not limited to the initial post-creation state.

As to (3), rollback queues of Steinman is not relied on in the interpretation. Applicant is arguing that the secondary reference cannot be bodily incorporated into the primary reference. The test for obviousness is not whether the features of one reference may be bodily incorporated into the other reference to produce the claimed subject matter but simply what the references make obvious to one of ordinary skill in the art. In fact, in comparative studies of state/context saving methods (full, infrequent and incremental state saving methods), the delta exchange/incremental/Steinman method has been proved to be more efficient overall than other methods. See reference to D. Bruce, "The Treatment of State in Optimistic Systems". The delta exchange/incremental/Steinman method imposes 5% overhead of processing time, while the full state saving method imposes 400%. (page 41, section 2.4). Therefore, Steinman improves scalability rather than teach away from it.

C. Regarding the Bishop reference, applicant argued (1) Bishop does not teach claim 3 because objects lack of strong references are destroyed during garbage collection, (2) Bishop does not teach the object provides any indication to the system that work is complete. (Page 6, 3rd para.).

The examiner disagrees. As to (1), Bishop provides two types of object references, strong references and weak references. While objects lack of strong references are destroyed during garbage collection, the weak references to objects are maintained. It is

the latter part of the teaching that is relied on in the interpretation. See rejection of claim 3 for detail.

As to (2), Bishop is not relied on to teach the object provides any indication to the system that work is complete, which is met by the other references relied on. See discussion of claim 1 for detail.

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Sue Lao whose telephone number is (703) 305-9657. A voice mail service is also available at this number. The fax phone numbers for the organization where this application or proceeding is assigned are (703) 308-9051 for regular communications and (703) 305-9731 for After Final communications.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-9600.

Sue Lao
December 14, 2000

MAJID A. BANANKHAH
PRIMARY EXAMINER